06 September 2025 09:59

Sliding window

networking → protocols

When to apply?

input → array or string
→ k

output → subarray or substring
→ max, min, sum, count, freq

sliding window → Fixed-size window
→ Variable-size window

Fixed Size window

maximum sum of k consecutive elements

arr → [2, 3, 1, 5, 7, 9, 4, 3, 2] k = 3

→ 5 + 7 + 9 = 21

→ Find subarray of size k whose sum is max.

0	1	2	3	4	5	6	7	8
2	3	1	5	7	9	4	3	2

1st window => 2, 3, 1 = 6

max → 6 9 13 21

→ o/p

	add	sub
sum = 6 + 5 = 11 - 2 = 9 =>	arr[3] ,	arr[0]
9 + 7 = 16 - 3 = 13	arr[4] ,	arr[1]
13 + 9 = 22 - 1 = 21	arr[5] ,	arr[2]
21 + 4 = 25 - 5 = 20	arr[6] ,	arr[3]
20 + 3 = 23 - 7 = 16	arr[7] ,	arr[4]
16 + 2 = 18 - 9 = 9	arr[8] ,	arr[5]

→ arr[i] , arr[i-k]

Sliding window of fixed size

Sliding window of fixed size

1st window calculate

// Grow
// Shrink

// Update my answer.

```
public class Main {
    public static void main(String[] args) {
        int[] arr = {2, 3, 1, 5, 7, 9, 4, 3, 2};
        int k = 3;
        System.out.println(maximumSum(arr, k));
    }

    public static int maximumSum(int[] arr, int k){
        int sum = 0;

        //1st window
        for(int i=0; i<k; i++){
            sum = sum + arr[i];
        }

        int ans = sum;

        for(int i=k; i<arr.length; i++){
            sum = sum + arr[i]; // Grow
            sum = sum - arr[i-k]; // Shrink

            ans = Math.max(ans, sum);
        }
        return ans;

    }
}
```

713. Subarray Product Less Than K

Solved

Medium | Topics | Companies | Hint

Given an array of integers `nums` and an integer `k`, return *the number of contiguous subarrays where the product of all the elements in the subarray is strictly less than* `k`.

Example 1:

```
Input: nums = [10,5,2,6], k = 100
Output: 8
Explanation: The 8 subarrays that have product less than 100 are:
[10], [5], [2], [6], [10, 5], [5, 2], [2, 6], [5, 2, 6]
Note that [10, 5, 2] is not included as the product of 100 is not strictly less than k.
```

arr → 1, 2, 1, 3, 4, 2

k : 10

prod : 2 4 12 4

```
loop( ) {
// grow

// shrink

// update answer
}
```

1 2 1 3 4 2

1 2 2 1 1 3 2 4

1 2 1 2 3

1 2 1 3

1 + 2 + 3 + 4 + 1 + 2 = 13

Grow → product

Shrink → divide

arr → 1, 2, 1, 3, 4, 2 (indices 0 1 2 3 4 5)

k : 10

prod : 2 4 12 4

grow => end++;

shrink => start++;

start → 0

end → 0 1 2 3 4

end : 3

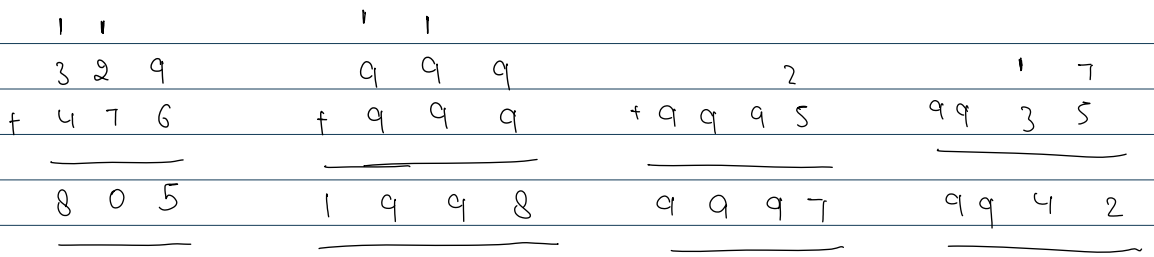
start : 0

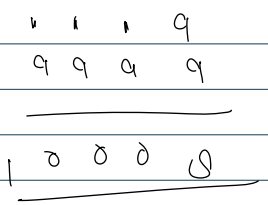
```
window : end - start + 1
```

arr → 1, 2, 1, 3, 4, 2 (indices 0 1 2 3 4 5)

```
public class Main {
    public static void main(String[] args) {
        int[] arr = {1, 2, 1, 3, 4, 2};
        int k = 10;
        System.out.println(productLessThanK(arr, k));
    }

    public static int productLessThanK(int[] arr, int k){
        int ans = 0;
        int start = 0;
        int end = 0;

        int prod = 1;

        while(end < arr.length && start<=end){
            //grow
            prod = prod * arr[end];

            //shrink
            while(prod >= k){
              prod = prod/arr[start];
              start++;
            }

            //update answer
            ans = ans + (end-start+1);

            end++;
        }
        return ans;
    }


}
class Solution {
    public int numSubarrayProductLessThanK(int[] arr, int k) {
        int ans = 0;
        int start = 0;
        int end = 0;

        int prod = 1;

        while(end < arr.length){
            //grow
            prod = prod * arr[end];

            //shrink
            while(prod >= k && start<=end){
              prod = prod/arr[start];
              start++;
```

```
            }

            //update answer
            ans = ans + (end-start+1);

            end++;
        }
        return ans;
    }
}
```

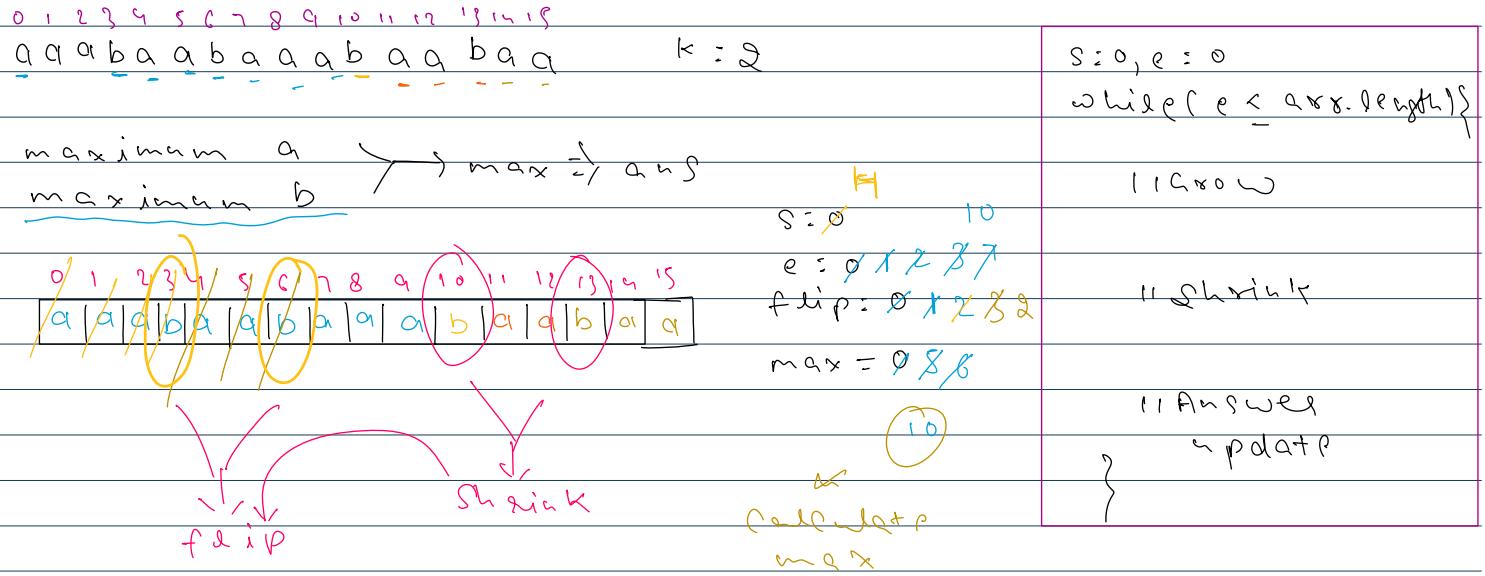





```
public class Main {
    public static void main(String[] args) {
        int[] arr1 = {1, 0, 2, 9};
        int[] arr2 = {3, 2, 4, 5, 6, 7};

        int[] ans = sumOfTwoArrays(arr1, arr2);

        for(int i=0; i<ans.length; i++){
            System.out.print(ans[i]+", ");
        }
        System.out.print("END");
    }

    public static int[] sumOfTwoArrays(int[] a1, int[] a2){
        int i = a1.length - 1;
        int j = a2.length - 1;

        ArrayList<Integer> list = new ArrayList<>();

        int carry = 0;

        while(i>=0 && j>=0){
            int sum = a1[i] + a2[j] + carry;
```

```
        if(sum>9){
            sum = sum%10;
            carry = 1;
        }
        else{
            carry = 0;
        }

        list.add(sum);

        i--;
        j--;
    }

    while(i>=0){
        int sum = a1[i] + carry;
        if(sum>9){
            sum = sum%10;
            carry = 1;
        }
        else{
            carry = 0;
        }
        list.add(sum);
        i--;
    }

    while(j>=0){
        int sum = a2[j] + carry;
        if(sum>9){
            sum = sum%10;
            carry = 1;
        }
        else{
            carry = 0;
        }
        list.add(sum);
        j--;
    }

    System.out.println(list);

    int[] ans = new int[list.size()];
    int l = 0;
    for(int  k=list.size()-1; k>=0; k--){
        ans[l] = list.get(k);
        l++;
    }
```

```
        return ans;
    }

}
```




```
public class Main {
    public static void main(String[] args) {
        int k = 2;
        String str = "abba";
        System.out.println(
            Math.max(
                maxLength(str, 'b', k),
                maxLength(str, 'a', k)
            )
        );
    }

    public static int maxLength(String s, char ch, int k){
        int start = 0, end = 0, flip = 0, ans = 0;

        while(end < s.length()){
            //Grow
            if(s.charAt(end) == ch){
                flip++;
            }

            //Shrink
            while(flip > k && start<=end){
                if(s.charAt(start) == ch){
                    flip--;
                }
```

```
                start++;
            }

            //Update answer
            ans = Math.max(ans, end-start+1);

            end++;
        }
        return ans;
    }

}
```